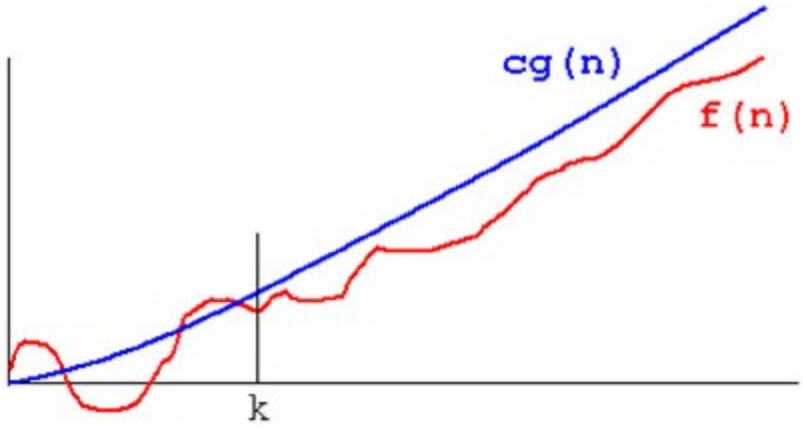
CS 288 2019S Section 102
Big-O Notation

Definition: A theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size n, which is usually the number of items. Informally, saying $f(n) = O(g(n))$ means $f(n)$ is less than some constant multiple of $g(n)$. The notation is read, "f of n is big oh of g of n".

Formal Definition: $f(n) = O(g(n))$ means there are positive constants c and k, such that $0 \leq f(n) \leq cg(n)$ for all $n \geq k$. The values of c and k must be fixed for the function f and must not depend on n.



As an example, $n^2 + 3n + 4$ is $O(n^2)$, since $n^2 + 3n + 4 < 2n^2$ for all $n > 10$ (and many smaller values of n). Strictly speaking, $3n + 4$ is $O(n^2)$, too, but big-O notation is often misused to mean "equal to" rather than "less than". The notion of "equal to" is expressed by $\Theta(n)$.

The importance of this measure can be seen in trying to decide whether an algorithm is adequate, but may just need a better implementation, or the algorithm will always be too slow on a big enough input. For instance, quicksort, which is $O(n \log n)$ on average, running on a small desktop computer can beat bubble sort, which is $O(n^2)$, running on a supercomputer if there are a lot of numbers to sort. To sort 1,000,000 numbers, the quicksort takes 20,000,000 steps on average, while the bubble sort takes 1,000,000,000,000 steps!

Time Complexity

In computer science, time complexity is the computational complexity that describes the amount of time it takes to run an algorithm. Algorithmic complexities are classified according to the type of function appearing in the big O notation.

The following table summarizes some classes of commonly encountered time complexities:

Big-O Notation	Complexity	Example
$O(1)$	constant	Determining if a binary number is odd or even
$O(\log n)$	logarithmic	Binary search
$O(n)$	linear	Finding the smallest element in an unsorted array
$O(n \log n)$	linearithmic	Fastest possible comparison-based sort
$O(n^2)$	quadratic	Selection sort
$O(n^3)$	cubic	Naive multiplication of two square matrices
$O(c^n)$	exponential	Decomposing a composite number into a product of integers (integer factorization)
$O(n!)$	factorial	Solving the traveling salesman problem via brute-force search

Array Sorting Algorithms Time Complexity

The following table describes integer sorting algorithms. Complexities assume n items to be sorted, with keys of length w:

Sorting Algorithm	Best	Average	Worst
Selection Sort	n^2	n^2	n^2
Insertion Sort	n	n^2	n^2
Bubble Sort	n	n^2	n^2
Quicksort	$n \log n$	$n \log n$	n^2
Mergesort	$n \log n$	$n \log n$	$n \log n$
Radix Sort	nw	nw	nw

Sources:

- Dictionary of Algorithms and Data Structures
- Wikipedia, the free encyclopedia